CHAPTER 2
Feature Extraction And Database Generation

2.1 Introduction

As a major factor influencing recognition performance, features play a very important role in handwriting recognition. This has led to the development of a variety of features for handwriting recognition and their recognition performances have been reported on standard databases [58, 107]. Some recent papers include those proposing directional distance features [143], gradient based features [132], wavelet-based features [91], pixel-distance features [142] and concavity features [31]. A feature is a measurement taken on the input pattern to be classified. Feature extraction refers to the process of finding a mapping that reduces the dimensionality of patterns. The features do not necessarily convey any intuitive meaning to a human and the dimensionality of the feature vectors is very high, in the hundreds. So it is difficult to understand their discriminative characteristics. A systematic evaluation of features in a specific feature vector is very important for designing a new feature vector by combining different feature vectors.

Feature extraction has been one of the most important topics in pattern recognition and has been studied by many authors [34, 151, 163]. Regarding the tool for evaluating features, a good tutorial on the criteria and selection of a good subset of features can be found in [75, 119, 128]. Node pruning for neural network classifiers [115], entropy measurement [40], and class separation [28] are some of the conventional methods. Using neural networks for feature extraction and data projection has been of great interest [16, 71, 81]. Mao and Jain presented extensive analysis in this area [81].

In the context of feature extraction for automatic character recognition, features were customarily defined as two types: global and local features. In global feature approach, several orthogonal transformations such as Fourier [105], Walsh [43] and
Karhunen–Loeve [126, 162] transforms are suggested. Among them, the K-L expansion has been proved to be a successful algorithm in machine printed Chinese character recognition. An experiment with a recognition rate of 94.6% was carried out by Kurosawa et al. [84]. For the recognition of handwritten Chinese characters. However this approach is complex and time consuming. In local features approach, a lot of methods which can be divided into four categories have been surveyed by Mori et al.[104].

2.2 Extraction of Features

The feature vector extraction is essential for efficient data representation and for extracting meaningful features for later processing. The selection of stable and representative set of features is the heart of a pattern recognition system. The aim of feature extraction is to divide the pattern by means of minimum number of features or attributes which are effective in discriminating between the different pattern classes. We define the feature as a measurement taken on the input pattern that is to be classified. Typically, we are looking for features that will provide a definite characteristic of that input type. The classifier is then supplied with a list of measured features, so that it maps these input features onto a classification state. That is, given the input features, the classifier must decide which type of class or category they match most closely. Classification is rarely performed using a single measurement or feature taken from the input pattern. Usually, several measurements are required to be able to adequately distinguish between inputs that belong to different classes. If we make ‘n’ measurements on our input pattern, each of which is a unique feature then we can use the algebraic notation to create a set of these features and call it as a feature vector. The dimensionality of the vector, that is the number of elements in it, if n, creates an ‘n’ dimensional feature space.
2.2.1 Conventional Feature Vector Extraction Method

The conventional method is to consider a region which includes the letter. If a line passes through a pixel, we give the corresponding pixel value one (1). Otherwise it is taken as zero (0) [26]. Thus, we have to store the pixel value combinations for every letter. Now, when an input pattern is given, a suitable match with these stored patterns is checked. Only if the new input pattern closely matches with any of the stored patterns, it will be identified. In case of English alphabets, there are different fonts for every character. The same letter written by different people and when written at different times will be different. Hence, a generalised condition for recognising a particular character cannot be specified, which is a must for the algorithmic approach. This is the major drawback of the conventional method – the inability to generalise. Another drawback is the large memory space required to store the pixel values.

2.2.2 Use of Neural Networks for Character Recognition

Artificial neural networks are used for achieving recognition because rather than programming them, we train the nets by examples. Programmers need not give the neural networks quantitative description of objects being recognised and sets of logical criteria to distinguish such objects from similar objects. Instead, we give examples of objects with their identification. The network memorizes this information by modifying the values in its weight matrix and will produce the correct response when the object is seen again [120].

If we use a neural network to recognize the characters from their pixel value combinations, we will get the needed generalization. But there will be a large number of units and accordingly, the weighted connections will also increase. This will result in a very complex network. One technique to reduce the complexity of the network and storage requirements is the bar mask encoding method [14].
2.3 Conventional Feature Extractor

The bar mask used in the experiment is similar to the seven-segment alpha numeric display. It converts the input characters into a highly compressed format suitable for recognition. In handwriting recognition, the symmetry of the character is one of the simplest features that can be extracted easily. Most of the characters are symmetrical about their horizontal and vertical axes. Therefore we can extract these features easily. However, some of the characters are symmetrical about their diagonal axes as well. These cross strokes can be detected by using the diagonal bars. Two additional vertical bars are used to take care of the vertical strokes as in 'I' and 'T' and four additional diagonal bars are used to take care of the cross strokes as in 'X' and 'N'. Thus the final design of the encoder consists of thirteen bars: Three horizontal, six vertical and four diagonal bars, which extract thirteen features [26].

However, this method is suitable only for the uppercase letters of the English alphabet. This cannot be used for the recognition of characters which are written by hand. One of the drawbacks of this method is that if the horizontal features are displaced (which is usual in case of handwritten numerals such as '8', '3' etc.) then they are not properly taken care of. This drawback is overcome in the modified feature extractor.

2.4 Modified Feature Extractor

The modified feature extraction method makes use of a 64 x 64 bit map as shown in the Fig.2.1. The bit map is divided into 3 horizontal bars (HF1, HF2, HF3, as in Fig. 2.1a), 3 vertical bars (VF1, VF2, VF3, as in Fig. 2.1b), 2 central bars (CF1, CF2, as in Fig. 2.1c), seven diagonal bars (DF1, DF2, DF3, DF4, DF5, DF6, DF7, as in Fig. 2.1d1-2.1d3). The diagonal bars DF1, DF2, ..., DF7 take care of cross strokes as in the letters 'N', 'X', 'W', '2' etc. The capture region associated with DF5 helps in distinguishing letter 'E', from 'F', '6' from 'C' etc. DF4 and DF6 help in distinguishing '4' from '9'. The vertical bars VF1 to VF4 take care of vertical strokes as in the letters 'I' and 'T'. The central bars CF1 and CF2 help in distinguishing character '3' from '8', '8' from '0'
etc. They also take care of displaced center features which normally occur with characters ‘8’, ‘3’, ‘S’, etc. The encoder converts the input character into a highly compressed format suitable for recognition. The character to be encoded is first standardized by scaling it so that it extends to the full height and width of the 15-segment encoder region. Fig. 2.2 shows the 15-segment feature encoder on which a sample ‘8’ is mapped. Each cell in the character is assigned a value.

For example: The ith row and jth column cell Cij is assigned a value Vcij where, Vcij = 1/(i*10+j). Associated with each segment is a capture region. The word ‘capture region’ is used to denote that the segment captures the values of all cells within its specified region. However, the feature value F,(say) associated with each region is given as

\[ F = \frac{\text{sum of values of ON cells}}{\text{sum of values of all cells}}. \]

2.5 Feature Extraction Algorithm

1. Read the pattern.
2. For each cell Cij = [ij] of the pattern assign a unique value VCij, where
   \[ VCij = 1/(i*10+j) \]
3. For each region, the feature value \( F = \frac{\text{sum of values of 'ON' cells}}{\text{sum of values of all the cells}}. \)
4. Repeat this for all the patterns.

Example: Feature value HF1 is calculated as follows:

```plaintext
for (i = 1 to 5)
{
    for (j = 1 to 15)
    {
        F[i][j]=1/(i *10 +j)
    }
} Numerator=Denominator=0;
```
Read the pixel matrix \( P[i][j] \) containing the pattern.

\[
\text{for}(i=1 \text{ to } 5)
\]
\[
\{ \\
\text{for}(j=1 \text{ to } 15)
\{ \\
\text{if}(P[i][j]==1) \\
\quad \text{Numerator} = \text{Numerator} + F[i][j] \\
\} \\
\text{Denominator} = \text{Denominator} + F[i][j]
\}
\]

\( HFI = \frac{\text{Numerator}}{\text{Denominator}} \)

The fifteen features so extracted are then given to the feature selector for selecting only the relevant features. The feature selector accepts the fifteen element feature vector \( \{F_1, F_2, \ldots, F_{15}\} \) as input and selects the ten best features which are then fed to the classifier.

2.6 Sample Generation

2.6.1 Sample Databases: Validation of the character recognition systems is usually carried out by running them blindly against independent test sets on which the systems have not been trained. For these tests to be conclusive, the validation sets should include a fairly large number of samples to reflect the variety of writing styles.

Many factors account for the diversity in handwriting styles, among them are the national or regional origin of the writers, their educational level, profession and age. Furthermore, for a given individual, different circumstances will affect his/her handwriting, such as stress, fatigue and hurry. Various technical aspects also play an important role in the quality of the digitised images: paper and ink colour, the kind of writing instrument, porosity of the paper etc. Many different writing styles are apparent, as well as numerals of different sizes and stroke widths.
It is important to realize that recognition systems cannot be compared simply by their reported performances, since most of the systems are still tested on the databases with very different characteristics. Therefore, other important factors must be taken into account when comparing the results such as; whether the test set is composed of the same number of samples for each class? Is there a need to normalize the results? Where does the data come from? Compared with the training set, were the samples in the testing set collected from different writers? Krzyzak et. al. [83] used CENPARMI datasets to develop a recognition system. It uses Fourier descriptors as dominant features and applies a modified back-propagation model for the classification of handwritten digits. Ahmed and Suen used back-propagation neural network for handwritten ZIP code recognition. The data sets used are unbalanced and information is lacking for normalization.

2.6.2 Sample Generation Algorithm: The performance of the classifier improves as the number of samples used to train the system increase. Our database consists of 10,000 samples each of size 64 x 64. The sample generation algorithm has been developed that takes as seed values, the samples collected from various research papers of Concordia University [22, 147]. These samples were originally selected from 17,000 sample database of U.S. Postal Services collected from various part of U.S.A. As it is very difficult to normally collect 10,000 samples we have developed an algorithm that uses a standard set of numerals and iteratively produces a large number of samples by distorting the original set of samples.

The database initially contains only the standard set of samples. A sample is chosen from this database and a path array (array of all marked bits) is obtained. The path array is a 2 dimensional array. Each row of the array represents a marked point, 2 columns are used to store the x and y co-ordinates. Each marked pixel is replaced by one of its 8 surrounding pixels. The selection is done in a random manner. The resulting path array represents a slightly distorted version of the original sample. The sample is reconstructed from the path array and is appended to the end of the database. Then the
next sample from the database is chosen and the process is repeated. Thus the samples so generated are further made use of to generate more distorted samples. The algorithm terminates when the required number of samples are generated.

2.7 Conclusion

The developed system is suitable for the feature extraction of off-line unconstrained handwritten characters. It transforms each two dimensional document handwritten character image into the corresponding one dimensional feature vector. Hence the developed model efficiently performs dimensionality reduction. It also reduces the volume of data to be processed. This feature extractor model has been used as the preprocessor for the neural network classifier model. The developed database model generates samples by taking as seed values, the samples from Concordia University handwritten numeral database. This database model can generate any required volume of off-line handwritten numerals.
Fig 2.1 Fifteen Segment Encoder
Fig. 2.1a Horizontal Regions of 15-Segment Encoder
Fig. 2.1b Vertical Regions of 15 Segment Encoder
Fig 2.1 c Central regions of 15-segment encoder
Fig. 2.1 d1 Diagonal Regions of 15 segment encoder
Fig. 2.1 d2 Diagonal Regions of 15 segment encoder
Fig. 2.1 d3 Diagonal Regions of 15-segment encoder
Fig 2.2 Fifteen Segment Encoder on which sample '8' is mapped